Engineer

SQLite

en hel databas i en enda fil

Vad det är

SQLite är en SQL-databasmotor som körs inuti programmet som använder den, som ett bibliotek, utan serverprocess, utan nätverksport och utan administratör, och som håller sina tabeller, index, triggers och vyer samlade i en vanlig fil på disk. Namnet dyker också upp som `sqlite3` : så heter kommandoradsskalet, och det är prefixet för varje funktion i dess C-gränssnitt. Den enda C-källfil som projektet levererar heter `sqlite3.c` . Projektets egen sammanfattning är att den är fristående, serverlös, kräver ingen konfiguration och är transaktionell, och projektet ber om att den jämförs med `fopen()` snarare än med Oracle. Det den ersätter är den hemsnickrade datafilen, inte databasservern.

Projektet började den 9 maj 2000, och koden är allmän egendom (public domain), så det finns ingen licens att klara av innan den levereras inuti en produkt. Filformatet är en del av uppgörelsen. En databas som skrivits på en 32-bitarsmaskin kan kopieras till en 64-bitarsmaskin, eller mellan processorer med big-endian och little-endian, och öppnas utan ändringar. Utvecklarna säger att de avser att stödja SQLite fram till år 2050, och kongressbiblioteket i USA, Library of Congress, listar dess filer som ett rekommenderat format för att bevara data. Med alla funktioner påslagna kan biblioteket vara mindre än 900 KiB, och därför sitter det i telefoner, bilar, klockor och skrivbordsprogram lika väl som på servrar.

Inuti är den en komplett SQL-motor: joins, delfrågor, vyer, triggers och transaktioner som förblir atomära, konsistenta, isolerade och beständiga även när strömmen går mitt i en skrivning. Två designval skiljer den från serverdatabaser. Det första gäller samtidighet. Hur många anslutningar som helst får läsa på samma gång, men bara en får skriva vid varje ögonblick, och låset omfattar hela filen. Det andra gäller typning. En kolumns deklarerade typ är en preferens, som SQLite kallar affinitet, och typen hör egentligen till varje värde, så en kolumn som deklarerats som `INTEGER` tar emot text som inte ser ut som ett tal.

Mot alternativen är gränsen tydlig. PostgreSQL, MySQL och SQL Server är delade tjänster som många maskiner talar med över ett nätverk. SQLite är ett privat lager för en applikation på en värddator. Jämfört med CSV- eller JSON-filer lägger den till ett frågespråk, index, villkor och transaktioner, till priset av ett beroende som de flesta språk redan har med sig.

Använd begreppet i ett beslut

Använd ”Vad är SQLite?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. En struktur som speglar hur verksamheten faktiskt fungerar, så att de frågor du behöver svar på senare fortfarande går att ställa.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Utformade och drev 6 ETL/ELT-pipelines — 6 ETL-pipelines över tre databasmotorer

Utformade och drev 6 ETL/ELT-pipelines Hör av er →

Varför det spelar roll

För en stor del av all mjukvara är den svåra frågan om en databas inte hur snabb den är, utan vem som driver den. En server måste installeras, uppgraderas, skyddas, övervakas och säkerhetskopieras, och någon måste få betalt för att göra allt det. SQLite tar bort hela den kategorin. Att öppna en databas är att öppna en fil, en säkerhetskopia är en kopia av den filen, tagen genom API:et för säkerhetskopiering medan filen används, och en testsvit kan ge varje test en egen ny databas.

Den ändrar också vad en fil kan vara. Ett program som annars skulle hitta på ett format, skriva en parser för det och hoppas att en krasch aldrig inträffar mitt i en sparning kan lagra sitt tillstånd i SQLite och få atomära sparningar på köpet. Projektet nämner den användningen direkt: som filformat för en applikation, som behållare för att flytta data mellan system, där mottagaren kör SQL i stället för att skriva en parser, och som en tillfällig databas för att sortera och sammanfatta data som kom i någon annan form.

Det beslut som spelar roll är snävt, och det går att pröva innan en enda kodrad skrivs: hur många processer skriver, och på hur många maskiner? Ett program på en värddator, som skriver i korta skurar, är hemmaplanen för SQLite, och gott om webbplatser med låg till måttlig trafik körs bekvämt på den. Flera servrar som skriver till samma data, eller en arbetsbörda där många användare skriver i varje ögonblick, hör hemma i en klient–serverdatabas. Storleken avgör sällan, eftersom en fil kan rymma upp till 281 terabyte. Antalet skrivare gör det.

Även på hemmaplan finns kostnader att väga. `ALTER TABLE` stöder bara att byta namn på en tabell eller en kolumn och att lägga till eller ta bort en kolumn, så de flesta andra ändringar av en tabells form innebär att bygga en ny tabell, kopiera över raderna och byta namnen. Främmande nycklar är avstängda tills varje anslutning slår på dem. Och den lösa typningen släpper in dåliga data i tysthet, om inte en tabell deklareras så att den vägrar ta emot dem.

Så ser det ut i praktiken

En enda fil utan server räcker i tre exempel här: som källa till varje dokument, som indata till ett bygge och som ett ställe att granska resultat på.

Det första exemplet är en generator för CV:n, referensblad, portföljer och personliga brev. I det dokumenterade utvecklingsskedet var källan en SQLite-databas med 23 tabeller, från resultat och företag till profiler och referenser, som en pipeline i 19 steg fyllde, från tom fil till färdig databas, med ett enda kommando. Varje steg kunde upprepas säkert, så pipelinen kunde köras mot en befintlig databas utan att någon rad dubblerades. Fyrtioen Python-moduler, sammanlagt 10 580 rader, renderade sex utdataformat från den enda filen, med exakt två beroenden vid körning. Ett dokument riktat till en viss sorts läsare var en databasfråga över resultaten, vald av en profil, snarare än en redigerad kopia. Sidan sade öppet vad det kostade: med alla fakta samlade i SQLite-filen kunde ett dokument bara tas fram genom generatorn.

Det andra exemplet är den här webbplatsens egen portfölj, som i det dokumenterade exportskedet genererades från samma databas. En exportör öppnade SQLite-filen i skrivskyddat läge och skrev 495 sidor på tre språk, med bara standardbiblioteket i Python, och utdata checkades in så att webbplatsen kunde byggas utan generatorns miljö. Sidadresserna lagrades som rader, en per adress och resultat, så att en omskriven mening inte längre flyttade en sida. Samma arbete dokumenterade fällan som beskrivs i nästa avsnitt: en jämförelse av en period mot en textkolumn matchade alla trettiotvå rader.

Det tredje är det minsta och det mest typiska. En pipeline som gjorde om råa avläsningar av el till förbrukning per timme och zon skrev sina resultat antingen som CSV eller till en lättviktig SQLite-databas för inspektion, och Python-versionen bearbetade en dags data på 34 ms och trettio dagar på 180 ms. En databas som används på det sättet kräver ingen installation alls: vem som helst med skalet `sqlite3` kan öppna filen och ställa frågor mot den.

Var det går fel

Det mesta besväret med SQLite kommer av att behandla den som en liten PostgreSQL. Den är en annan maskin med andra kanter.

Ett filter på årtalet matchar varje rad. SQLite konverterar ett värde före en jämförelse bara när den ena sidan bär en affinitet. Ett funktionsresultat som det från `substr()` bär ingen, och det gör inte heller ett naket tal, så SQLite jämför texten `'2019'` med heltalet `2025` som de är, och i dess sorteringsordning kommer varje tal före varje textstycke:

```
CREATE TABLE stint (period TEXT, hours TEXT);
INSERT INTO stint VALUES
  ('2019-03', '9'), ('2024-11', '10'), ('2026-01', '42');

SELECT count(*) FROM stint
WHERE substr(period, 1, 4) >= 2025;                    -- 3

SELECT count(*) FROM stint
WHERE CAST(substr(period, 1, 4) AS INTEGER) >= 2025;   -- 1
```

Konvertera uttryckligen överallt där text möter ett tal. För nya tabeller deklarerar du riktiga typer och lägger till nyckelordet `STRICT`, som finns sedan version 3.37.0: en strikt `INTEGER`-kolumn tar fortfarande emot `'42'` och lagrar det som ett tal, och den avvisar `'ten'` med ett fel i stället för att lagra det.

Skrivningar får svaret ”database is locked”. En anslutning håller skrivlåset, och en andra skrivare väntar antingen eller ger upp direkt. Orsaken är oftast en lång transaktion, eller en läsare i standardläget med återställningsjournal som håller upp en skrivare. Ställ om databasen till write-ahead logging (WAL), där läsare och skrivaren slutar blockera varandra, håll skrivtransaktionerna korta och sätt en väntetid för upptagen databas så att en skrivare väntar en stund i stället för att ge ett fel. Om skrivarna verkligen är många och ständiga är det signalen att flytta till en server.

Filen blir korrupt på en delad disk. SQLite förlitar sig på operativsystemets fillås, och många nätverksfilsystem implementerar de låsen med buggar, så två maskiner kan till slut skriva till samma del av filen samtidigt. Write-ahead logging fungerar över huvud taget inte över ett nätverksfilsystem, eftersom varje

process som använder databasen måste dela minne på en värddator. Håll databasen på en disk som är lokal för det enda program som skriver till den, och sätt en server framför den om andra maskiner behöver datan.

En rad lever längre än föräldern den pekar på. Villkor för främmande nycklar deklareras i databasschemat men förblir inaktiva tills varje anslutning kör `PRAGMA foreign_keys = ON`, så en radering i en anslutning som hoppade över pragman lämnar föräldralösa rader efter sig utan protester. Sätt pragman där applikationen öppnar en anslutning, på det enda stället som gör det, och testa att den är påslagen.

Läs mer

- About SQLite — projektets egen beskrivning: körs i processen, serverlös, allmän egendom, en enda fil som fungerar på alla plattformar.
- Datatypes In SQLite — lagringsklasser, typaffinitet och jämförelsereglerna bakom fällan ovan.
- STRICT Tables — hur du får en tabell att upprätthålla sina deklarerade kolumntyper.
- Write-Ahead Logging — läsare och en skrivare parallellt, och varför det stannar på en värddator.
- Appropriate Uses For SQLite — var den passar, var en klient–serverdatabas passar bättre, och en checklista.
- SQLite Foreign Key Support — varför villkoren är avstängda som standard och hur du slår på dem.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med ett representativt dataset, frågorna det ska besvara och dagens flaskhals. Vi kan avtala mått på kvalitet, genomströmning och återställning.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är SQLite?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/sqlite/